Representing Many-Body Quantum States using Probabilistic Bits

Supplementary Material

January 26, 2024

In this document we provide background information to the main report. The information presented here serves to aid and enrich the understanding of the main report but is by no means necessary for understanding the main report. The topics also provide a broader context and motivation for our presented work and might be useful for future work in this topic.

Contents

1 Quantum Mechanics 101

This section aims to provide background information about quantum mechanics in general. It is written intentionally in a more general, abstract manner. It provides a broader overview of where neural network quantum states fit within the theory of quantum mechanics and why one would want to use them.

1.1 The Basics

In Quantum Mechanics (QM), the wavefunction $|\psi\rangle$ is a complex-valued function that contains all possible information about a quantum system. This encompasses the probabilities of the system being in various states, including the types of particles, their arrangements, and interactions. The wavefunction is not just a single state but a complete description of all possible states of the system.

The energy of a quantum system is defined through the Hamiltonian, an operator symbolized as $\hat{H}$. For non-relativistic and time-independent systems, the Hamiltonian acts on the wavefunction and returns the energy of the state, as expressed by the time-independent Schrödinger equation:

$$\hat{H}|\psi\rangle = E|\psi\rangle \tag{1}$$

This equation is key in determining the behaviour of a quantum system.

1.2 Representations

The abstract nature of QM allows for multiple mathematical frameworks or representations to describe the same physical phenomena. Each representation provides a unique viewpoint and set of tools for solving a quantum mechanical problem. For relatively simple systems, it is possible to keep $|\psi\rangle$ and $\hat{H}$ as abstract objects with a basis and analytically solve equation (1) using ordinary differential equation methods. The details of this can be found in standard entry-level QM books such as [1].

In the matrix representation, the wavefunction $|\psi\rangle$ is represented as a vector within a specific complex vector space, often referred to as Hilbert space. This vectors Hermitian conjugate (i.e. transposed conjugate) is denotes as $\langle\psi|$. The Hamiltonian $\hat{H}$ is represented as a matrix in this vector space. When this matrix acts upon a wavefunction (vector), it operates on the basis states of the system.

A wavefunction $|\psi\rangle$ can be expanded in terms of a chosen basis. For example, if you choose an abstract state basis $|s\rangle$, the wavefunction $|\psi\rangle$ can be represented as $|\psi\rangle = \int \psi(s)|s\rangle ds$, where $\psi(s)$ is the wavefunction in state space. Likewise, the wavefunction of a state in a particular basis is represented by the projection of the wavefunction onto the basis vectors, $\psi(s) = \langle s|\psi\rangle$.

1.3 Expectation Values

We can rewrite equation (1) by multiplying both sides by $\langle\psi|$:

$$E = \langle\psi|\hat{H}|\psi\rangle = \langle H\rangle \tag{2}$$

Where in the end we introduced the general shorthand notation $\langle . \rangle$ for the expectation value.

In quantum mechanics, observables (i.e. things you want to measure of your system such as energy, momentum or spin orientation) are defined as operators. This is because quantum states are represented as vectors in a complex vector space (Hilbert space), and the operations on these vectors are most conveniently expressed in terms of matrices. For example,

multiplying a vector (state) by a matrix (observable) results in another vector, representing the transformed state.

This procedure for calculating E works the same for any expectation value O we want to determine. In computational terms, equation 2 represents the operation of multiplying a transposed vector by a matrix and then by the original vector to obtain the energy of a quantum system:

$$O = (\bar{\mathbf{v}})^T \hat{O} \mathbf{v}$$

with $\mathbf{v}$ a column vector representing the wavefunction and $\hat{O}$ a matrix representing the observable

1.4 State Probabilities

Quantum mechanics is probabilistic by nature, meaning that we deal with probabilities of the system being in certain states. The absolute square of the wavefunction $|\psi(s)|^2$ gives the probability of finding the system in a particular state s. The overall probability distribution of the quantum system over all states is given by $|\psi|^2$. Because of this property, the wavefunction is also referred to as the 'quantum state' of a system. Note that this refers to *all* possible states and their probabilities, rather than the system being in one specific state s.

1.5 Exact Diagonalization

Using the matrix representation of QM, we can solve equation (1) using a method called exact diagonalization. Here, the Hamiltonian matrix is diagonalized using numerical algorithms. Diagonalization means finding a basis $|s\rangle$ in which the Hamiltonian matrix is diagonal. The diagonal elements of this matrix are the eigenvalues λ, and they represent the energy levels of the system. The corresponding eigenvectors represent the quantum states (wavefunctions) of the system at these energy levels.

The dimensionality of the Hilbert space scales exponentially in the amount of particles. For instance, if each particle can be in one of M states, then the Hilbert space for N particles has M^N dimensions. This exponential growth is why exact diagonalization becomes intractable and we have to refer to approximation methods to estimate $|\psi\rangle$. We will focus on a method called the variational method.

1.6 Variational Method

The variational method involves choosing a trial wavefunction, or 'ansatz', that is parametrized by some unknown parameters, and then adjusting these parameters to minimize the energy of the system. The wavefunction you obtain for the optimal set of parameters is an approximation of the ground state of your wavefunction.

The trial wavefunction is often based on some physical intuition or technical knowledge about the system. Take for example the trial wavefunction for determining the ground state of a Helium atom (2 electrons orbiting a nucleus) given by [2]:

$$\psi(r_1, r_2) = Ae^{Z(r_1+r_2)}$$

Where A is some proportionally constant, r_i is the distance of the i-th electron from the nucleus and Z is the tuneable charge of the nucleus. This model is based on the physical intuition that each electron perceives the nuclear charge as being "shielded" by the other electron. Therefore, an tuneable "effective" nuclear charge Z less than 2 is used in the trial wavefunction to reflect this interaction.

With this method, we essentially substitute the hard problem of finding an exact solution for two lesser hard problems:

1. Find a suitable ansatz for your wavefunction
2. Minimize the energy of your system w.r.t this ansatz

Note that this method reduces the complexity of the problem to finding a polynomial number of parameters instead of diagonalizing a massive Hamiltonian matrix.

1.7 The Ansatz Problem

Human-constructed wave functions have the advantage of being relatively easy to interpret, but they have the disadvantage of being difficult to systematically improve the accuracy of our ground state estimates for complex quantum systems. It is highly non-trivial to estimate the parametrization of the wavefunction for these kinds of systems and bias due to human insight may lead to qualitatively incorrect predictions.

This has inspired the use of neural networks to not only find the minimum energy solution for a specific ansatz but also to help in formulating a suitable ansatz itself. This method exploits the universal approximation capabilities and flexible representations of neural networks. These representations of the wave function are referred to as neural network quantum states and are described in the main body.

2 Heisenberg Model

In this section, we will explain the basics of Ising models and detail the Heisenberg model.

2.1 Ising models

In Ising models, particles are confined to a lattice, (physically due to the Coulomb force). Their positions are restricted to discrete points, leaving no variations in position and momentum. The only degrees of freedom then become the states at each lattice site. The size of the Hilbert space is now solely determined by the orientations of the spins at the lattice sites.

Classical spins can be thought of as small magnets with orientations that influence each other. A spin model takes into account the orientation, alignment and interactions of these spins, which are critical in determining the magnetic properties of the material.

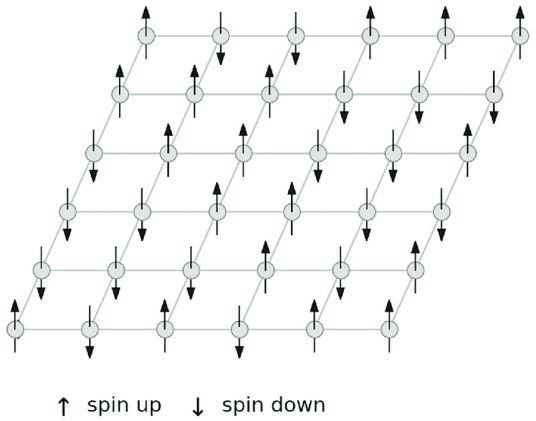

Figure 1: Visualization of an Ising model. Spins are arranged on a lattice and can be in two states, up or down. The way these spins interact with each other is determined by the Hamiltonian function. Figure taken from [3]

In spin-1/2 models, each spin can only point up or down (i.e. binary values). The full configuration of spins, either pointing up or down, is a single state of the system. How one spin affects another is modelled by the interaction strength, W_{ij}, and a spin's own energy change due to nearby interactions is shown by a term called b_i. A classical spin model is shown in Figure 1.

Note the resemblance of this system to that of neural networks: binary units interacting with each other through weighted interactions. The main two differences between Ising systems and ANNs is that in physical systems, spins often only influence nearby spins and their interaction strengths W_{ij} and b_i are fixed, while in ANNs there is no notion of distance and any node can be connected to any other node and the parameters are variable.

This system has a correspondence to a Boltzmann machine, where binary units interact with each other. The main differences are that in spin systems, spins usually only affect nearby spins and their interaction strengths W and b don't change, while in Boltzmann Machines any unit can connect to any other and the parameters W and b can change.

2.2 Quantum Spins

Quantum spin is a distinctive property of quantum particles with no direct classical equivalent. They cannot be visualized as tiny magnets pointing in a specific direction. They have an orientation, but it's conceptualized differently from the classical notion of direction.

A key aspect of quantum spins is that they can be in a state of superposition, meaning they can be in multiple states simultaneously. This is analogues to how qubits work (in theory, quantum spins can be used to represent quantum bits). This has the consequence that the exact orientation of a quantum spin is not determined until it is measured.

Once a spin state is measured, it's found to be aligned along the direction of measurement. This means that the 'direction' of a quantum spin depends on how you measure it. In other words, how you choose to measure the spin determines what information you'll get about it.

This happens because of the Heisenberg Uncertainty Principle. This principle tells us that you can't know all the properties of a quantum spin at the same time with perfect accuracy. For example, if a particle's spin is well-defined in the z-direction, its spin in the x and y directions is completely undefined. This is a big difference from classical spins, like a spinning top, which can have a specific direction and strength that you can measure any time.

When measuring quantum spin along the z-direction (or the sigma-z basis), the outcome is binary, either +1 or -1, analogue to classical binary spins. However, the underlying physics of quantum spins introduces quantum correlations in addition to classical ones. Even after a measurement, these correlations can persist. For example, in a spin chain, measuring one spin can instantaneously affect the state of another, entangled spin. This is a form of quantum information that can survive the measurement process.

These quantum correlations are essential for understanding the complex dynamics of quantum systems and are modelled using the hidden units of the Restricted Boltzmann Machine, as described in the main text.

2.3 Pauli Spin Matrices

The Pauli spin matrices are used to describe spin 1/2 systems and act as operators on the individual spins. $\hat{\sigma_i}$ represents the spin operator along the i-axis. The eigenvalues of these matrices correspond to the possible measurement outcomes of the spin in their respective directions. For a spin-1/2 particle, these eigenvalues are +1 and -1, corresponding to spin "up" and "down" along that axis.

In the sigma-z basis, the eigenvectors of $\hat{\sigma}^z$ are denoted as $|s_i = \pm 1\rangle$ with eigenvalues $s_i = \pm 1$. The operation of the Pauli spin operators on a single spin $|s_i\rangle$ are given by:

$$\hat{\sigma}_i^x|s_i\rangle = |-s_i\rangle \qquad \hat{\sigma}_i^y|s_i\rangle = is_i|-s_i\rangle \qquad \hat{\sigma}_i^z|s_i\rangle = s_i|s_i\rangle \tag{3}$$

We see that in this basis, the $\hat{\sigma}_i^x$ essentially flips the spin, $\hat{\sigma}_i^y$ flips the spin and multiplies it by a factor is_i and $\hat{\sigma}_i^z$ return the same state with a factor s_i.

In the sigma-z basis, the spin orientations can be represented using the vectors $|s_i = 1\rangle = (1,0)^T$ and $|s_i = 1\rangle = (1,0)^T$. Substituting this into equation (3), we find explicit matrix expressions for the Pauli spin matrices:

$$\hat{\sigma}^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \hat{\sigma}^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \hat{\sigma}^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{4}$$

In QM language, we would say that in the sigma-z basis only $\hat{\sigma}_i^z$ acts as a good observable. It acts on the state and returns the value we want to measure (the orientation of the spin along the z-axis) without altering the state. The other two Pauli spin matrices inherently alter the spin state when we try to measure its state.

2.4 Hilbert space

The basis of Hilbert space is a tensor product of the basis vectors $|s_i = \pm 1\rangle$ denoted as $|\sigma\rangle = |\sigma_1\rangle \otimes |\sigma_2\rangle \otimes \cdots \otimes |\sigma_N\rangle$. In this basis, the Hamiltonian becomes an operator with a

dimension equal to the square of the number of possible states $2^N \times 2^N$.

For two spins, this becomes: These elements are given by the tensor products of the Pauli matrices:

$$\sigma_1^x \otimes \sigma_2^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ 1\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

In a similar manner, other tensor products of Pauli matrices can be computed

2.5 The Heisenberg Model

The Heisenberg model is a crucial spin model in physics, widely used to describe quantum magnetism [4], [5]. It consists of a lattice of N quantum spins $s = \{s_1, \ldots, s_N\}$, where each spin can take only binary values $s_i = \pm 1$. The wavefunction $|\psi\rangle$ is expressed in the basis of all 2^N spin states $|\psi(s)\rangle$.

The Hamiltonian of the Heisenberg model is given by:

$$\hat{H} = -\frac{1}{2} \sum_{j=1}^{N} \left(J_x \hat{\sigma}_j^x \hat{\sigma}_{j+1}^x + J_y \hat{\sigma}_j^y \hat{\sigma}_{j+1}^y + J_z \hat{\sigma}_j^z \hat{\sigma}_{j+1}^z + h \hat{\sigma}_j^z \right) \tag{5}$$

The terms $J_x \hat{\sigma}_j^x \hat{\sigma}_{j+1}^x$, $J_y \hat{\sigma}_j^y \hat{\sigma}_{j+1}^y$, and $J_z \hat{\sigma}_j^z \hat{\sigma}_{j+1}^z$ represent the interactions between adjacent spins in the x, y, and z directions, respectively. The constants J_x, J_y, and J_z are the coupling constants for these spin interactions, influencing their strength and nature (ferromagnetic or antiferromagnetic).

The term $h \hat{\sigma}_j^z$ accounts for the influence of an external magnetic field on the spins, where h is the field's strength. This term describes how the spins align or oppose the external field.

This Hamiltonian described all the interactions of the quantum system. The balance between these elements determines the overall energy state of the spin system.

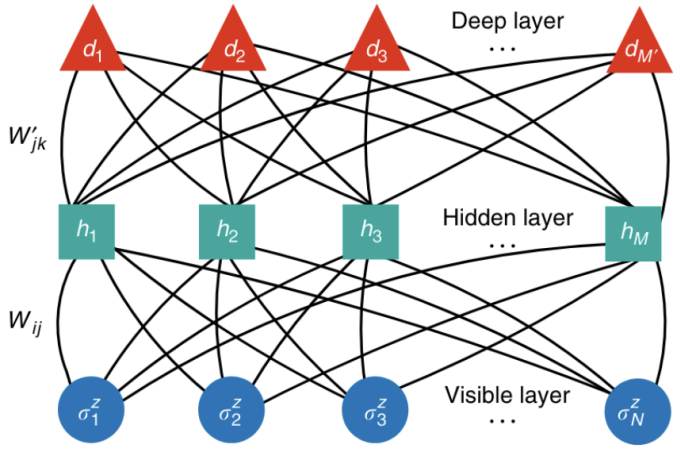

Figure 2: Illustration of the DBM network architecture. Figure taken from [7]

3 Deep Boltzmann Machines

Deep Boltzmann Machines (DBMs) are a more advanced form of RBMs. DBMs include an additional layer of hidden units, known as deep units. This extra layer allows DBMs to better represent complex quantum states [6]. The architecture of a DBM includes N input units σ, M hidden units h, M' deep units d, and a set of couplings and bias terms $\mathcal{W} = (a, b, b', W, W')$, as shown in Figure 2.

The probability of a configuration $\mathcal{S}$ in the DBM is given by:

$$p(\mathcal{S}, \mathcal{W}) = \frac{1}{Z} \sum_{\{h,d\}} \exp\left[\sum_i a_i \sigma_i^z + \sum_{ij} \sigma_i^z W_{ij} h_j + \sum_j b_j h_j + \sum_{jk} h_j d_k W'_{jk} + \sum_k b'_k d_k\right] \quad (6)$$

Note that there is no efficient analytical formula for $p(\mathcal{S}, \mathcal{W})$. Because of this, numerical methods such as the Monte Carlo method are required to obtain the $p(\mathcal{S}, \mathcal{W})$ value, making the DBM more costly.

There are two primary methods for training DBMs. The first involves the same variational learning method as for RBMs The second involves a quantum-classical mapping to represent the wavefunction exactly with a neural network.

3.1 Variational DBM Learning

The variational learning of a DBM is very similar to that of an RBM. We can use the functional form of the DBM to form a variational ansatz:

$$\psi_M(\mathcal{S}, \mathcal{W}) = \sum_{\{h,d\}} \exp\left[\sum_i a_i \sigma_i^z + \sum_{ij} \sigma_i^z W_{ij} h_j + \sum_j b_j h_j + \sum_{jk} h_j d_k W'_{jk} + \sum_k b'_k d_k\right] \quad (7)$$

We aim to find parameters $\mathcal{W} = (a, b, b', W, W')$ that minimize the system's quantum energy. The input layer of the DBM represents 'quantum bits' as classical binary units. The hidden and deep units h_i learn to represent quantum behaviour. The advantage of deep units lies in their increased representational capacity, which allows for modelling a broader spectrum of quantum correlations. The optimal solution can again be found using the Stochastic Reconfiguration method.

Deep Boltzmann Machine Sampling

The restricted connectivity of RBMs allows for more efficient computation using CPUs due to the structure of the network, in contrast to the case of DBMs, where the hidden layers add complexity and thus are more computationally costly. Furthermore the optimization takes place over a larger set of parameters $W = (a, b, b', W, W')$ and that the expectation values required to update the parameters are harder to compute analytically. This increases the difficulty in implementing the DBM as opposed to RBM in classical CPU computation.

For the DBM, it is more convenient to take the Monte Carlo average over the marginal distribution $\tilde{w}(\sigma, \sigma'; d)$ instead of the full distribution. We do this by tracing out the h spins:

$$\tilde{w}(\sigma, ; d) = \sum_{h_1, h_2} \psi^\dagger(\sigma; h_1, d_1)\psi(\sigma; h_2, d_2) = \psi^\dagger(\sigma; d_1)\psi(\sigma; d_2), \tag{8}$$

with

$$\tilde{\phi}(\sigma; d) = \prod_j 2\cosh\left[b_j + \sum_i (W_{ji}\sigma_i + \sum_k W'_{jk} d_k\right]. \tag{9}$$

The formula for the expectation value $\langle \mathcal{O} \rangle$ can then be rewritten as:

$$\langle \mathcal{O} \rangle = \frac{\sum_{\sigma, h'} \widetilde{w}(\sigma, h', d')\mathcal{O}_{loc}(\sigma, h', d')}{\sum_{\sigma, h'} \widetilde{w}(\sigma, h', d')}, \tag{10}$$

where the local observable $\mathcal{O}_{loc}(\sigma, h', d')$ is given by

$$\mathcal{O}_{loc}(\sigma, h', d') = \frac{1}{2} \sum_{\langle \sigma \rangle} \left(\langle \sigma | \mathcal{O} | \phi^*(\sigma, h', d_1) \rangle + \langle \mathcal{O} | \phi^*(\sigma, h', d_2) \rangle \right). \tag{11}$$

This requires MH sampling over two variables, the visible and the hidden layers.

3.2 Quantum-to-classical Mapping

Adding a deep layer to the DBM network enables a unique method of representing quantum systems, which is not possible with RBMs. This method, known as quantum-classical mapping, involves representing the quantum wavefunction $|\psi\rangle$ exactly in a classical framework (the DBM network), without using machine learning techniques. Carleo et al. (2018) first introduced this method.

The central concept here is that a quantum system undergoing imaginary time evolution will naturally reach its lowest energy state, just like any physical system. The DBM wave function can accurately mimic the imaginary-time evolution of a Hamiltonian by dynamically changing its parameters (such as adding or removing nodes). The ground state parameters for the DBM network can be analytically derived. There's a direct link between the imaginary-time evolution of the quantum system and the updating of the DBM parameters.

To evolve an initial quantum state $|\psi_0\rangle$ through a time step τ, we use the matrix exponential of the Hamiltonian $\hat{H}$:

$$\psi(\tau) = e^{-\tau \hat{H}} |\psi_0\rangle \tag{12}$$

(ΔE^{-1}) between the ground and first excited state. For a large enough τ, which is significantly greater than the inverse of the energy gap Here, ΔE is that energy gap. For finite systems, the energy gap is usually finite, and the total time needed to reach the ground state accurately grows polynomially with system size [8].

Since the operators within $\hat{H}$ often don't commute (meaning their application order matters), the matrix exponential is non-trivial. We use Suzuki-Trotter decomposition to simplify this,

breaking down the exponential of a sum of non-commuting operators into a product of exponentials of individual operators. This approach uses the 2nd-order Trotter formula to approximate the calculation accurately:

$$\psi(\tau) = \mathcal{G}_1(\delta_\tau/2)\mathcal{G}_2(\delta_\tau)\dots\mathcal{G}_1(\delta_\tau)\mathcal{G}_2(\delta_\tau)\mathcal{G}_1(\delta_\tau/2)|\psi_0\rangle, \tag{13}$$

This decomposition breaks the Hamiltonian into two parts, $\hat{H} = \hat{H}_1 + \hat{H}_2$, and uses short-time propagators $\mathcal{G}_\nu(\delta) = e^{-\hat{H}_\nu \delta_\tau}$ and $\delta_\tau = \frac{\delta}{N_t}$.

The challenge is to find a neural network representation of ψ such that after applying these propagators, we obtain the DBM with new parameters $\tilde{W}$:

$$e^{-h_\nu \delta_\tau}|\psi_W\rangle = C|\psi_{\tilde{W}}\rangle. \tag{14}$$

Applying time evolution to the quantum system means changing the parameters $W \to \tilde{W}$, by altering existing parameters or adding new ones[1].

Since each Hamiltonian requires a different decomposition, the construction rules vary accordingly. Repeatedly applying these transformations creates a DBM representing the ground state of the system. The challenge remains in obtaining samples from the DBM after this process. For specific implementations, we refer to Carleo et. al (2018) [8].

[1]This method is similar to the path-integral formalism in quantum systems, first introduced by Richard Feynman [9].

4 Stochastic Ising Machines

Mainstream ML algorithms are designed and chosen with CPU implementation in mind. Hence, some models are heavily preferred over others even though they are less powerful theoretically. Despite being powerful models, Boltzmann machines and their variants fell out of favor from mainstream deep learning praxis primarily because they are computationally hard to train with conventional CPU and GPU hardware [10]. In this section we will describe how a stochastic Ising machine could provide a hardware acceleration for these energy-based models, similar to how the GPU provided a hardware acceleration for artificial neural networks. We will also detail the practical aspects of building such hardware system.

4.1 Ising Machines

An Ising machine is a hardware implementation of the Ising model, which can be used find the minimum energy configuration of a spin system, which corresponds to the optimal solution of a given problem. Ising machines can exploit the natural dynamics of a physical system to find low-energy states and they can potentially be more energy-efficient than traditional electronic computers when it comes to solving certain optimization problems.

There are many different types of Ising machines, each using a different physical representation of an Ising model [11]–[13]. In these Ising machines, the "spin" in the Ising model is abstracted to some physical system that can emulate the necessary behavior of the binary states. By changing physical interactions in the system, the parameters of the Ising Hamiltonian can also be manipulated.

4.2 Stochastic Ising Machines

A stochastic Ising machine is a an Ising machine implemented with p-bits. Stochastic Ising Machines are particularly useful in probabilistic algorithms like Monte Carlo methods. They exploit the natural mapping between the intrinsically noisy physics of nanomagnets to the mathematics of general probabilistic algorithms (e.g., Monte Carlo, Markov Chain Monte Carlo).

Generating pseudo-random numbers typically requires thousands of transistors [14], while p-bits enable the creation of true random number generators (RNGs) with just three transistors. Just as quantum computations are more efficient on quantum computers with qubits, probabilistic computations might be more apt for probabilistic computers using p-bits. This is reflected by the computational complexity of the SR method for the deep Boltzmann machine, described in the main text. For this computation, the main advantage of using a stochastic Ising machine is to replace the expensive (double) MCMC computations with samples from the stochastic Ising machine, which can provide orders of magnitude improvement over commonly used software implementations [15].

Just as quantum computations are more efficient on quantum computers with qubits, probabilistic computations might be more apt for probabilistic computers using p-bits. Moreover, as the use of quantum computers is often impractical, due to the necessity of using cryogenic operating temperatures and the vulnerability to noise, it is often beneficial to simulate the quantum system on classical computers, which however can be very time-consuming and limited to small systems. Thus, stochastic Ising machines could provide a room-temperature solution to boost the simulation speed and potentially enable the simulation of large-scale quantum systems [16].

4.3 Sparsity

Network topologies such as RBMs and DBMs are highly impractical to implement in hardware due to the all-to-all interlayer connections between the nodes. The number of connec-

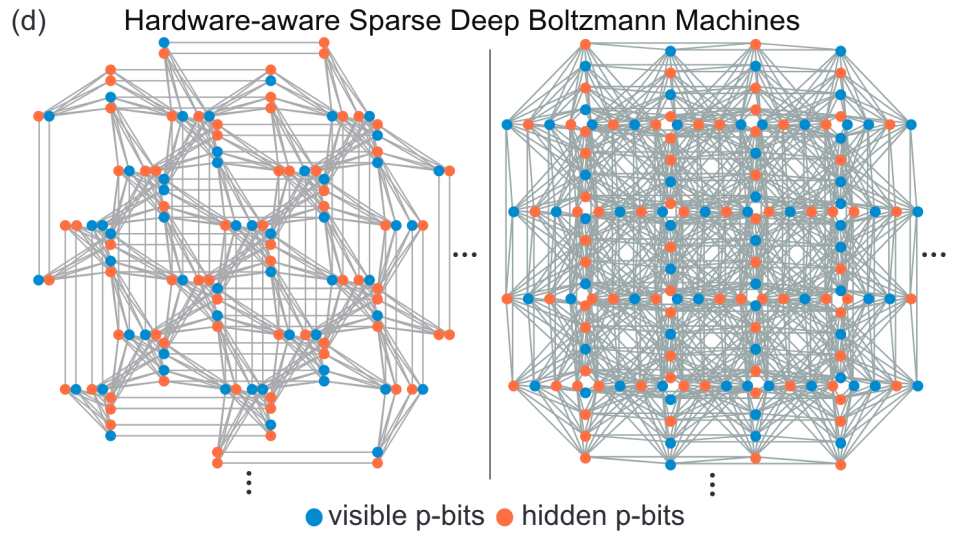

Figure 3: Hardware-aware sparse Deep Boltzmann Machines (DBMs) can be represented with visible and hidden p-bits through a minor graph embedding to the Pegasus (left) and Zephyr (right) graphs. Figure from [15].

tions grows quadratically with the number of nodes, leading to a potentially unmanageable number of connections as the network size increases. Physical hardware has limitations on how many connections can be effectively implemented [15]. Furthermore, as we will explain later, dense networks don't lend themselves well to highly parallel operation.

To address these issues, we need to map the dense network architectures onto a sparse graph, which has fewer connections, through a process known as minor graph embedding [17]. This process finds a way to represent the densely connected network within a sparsely connected graph without losing the ability to perform the necessary computations[2]).

To convert RBM and DBM graphs into sparse graphs without losing functionality, a technique known as minor graph embedding can be used. Sparse RBMs differ from standard ones in that they lack fully-connected interlayer links. However, they allow intra-layer connections, enhancing the network's representational capacity.

Minor graph embedding of a RBM or DBM network can be done by randomly allocating visible and hidden bits within Pegasus or Zephyr graphs of different sizes (see Figure 3) [19]. These hardware-aware networks limit the connectivity to a maximum of 15 to 20 connections per node. It's important to note that this embedding may appear limiting at first glance, but it's actually advantageous. The network's massively parallel architecture and the inherent MCMC sampling using p-bits enable direct training on the full network.

4.4 Accelerated Gibbs Sampling

Color Gibbs sampling is a technique used to accelerate Gibbs sampling for digital p-bit networks. It exploits the fact that for undirected networks (i.e. symmetric matrix W), it does not matter in which order the p-bits are updated, as long as they are not updated at the same time. This means that while connected p-bits must be updated one after the other, if two p-bits are not directly linked, updating one does not immediately affect the other.

[2]Interestingly, the human brain also is a sparse network, with each neuron having, on average, only 1000 of the $8.6 * 10^9$ possible connections to other neurons [18].

Furthermore, if the underlying graph is *sparse*, it is often easy to split it into disconnected chunks by coloring the graph using a few colors. This approach involves assigning different colors to connected p-bits and the same color to unconnected p-bits. Each color block is then updated at the same time to exploit parallel updating of unconnected p-bits.

Finding the minimum number of colors is an NP-hard problem [20] but heuristic methods like Dsatur, which are less complex, can color graphs fast without always getting the fewest colors. For sparse graphs, where getting the exact minimum isn't crucial, only a few colors are usually needed.

Hardware p-bit networks, as detailed in the next section, have the possibility to implement truly asynchronous Gibbs sampling. This means that the p-bits update in a truly random fashion, independent of each others clock. In such scenario, the chance of a parallel update is low. In such a system, there is no need for graph coloring or phase-shift engineering, but we retain the massive parallelism [16].

4.5 Hardware P-bits

Much of the current literature emphasizes proof-of-concept implementations of digital p-bits, primarily utilizing field-programmable gate arrays (FPGAs). These FPGA-based implementations offer relatively easy implementation and are able to outperform conventional hardware in Gibbs sampling tasks, as shown in Table 1 and Table 2. However, they have limited scalability to real-world applications because they require a large number of transistors per digital p-bit and have a large digital footprint [21].

Probabilistic bits can also be physically implemented using analog mixed-signal nanodevices [16]. In theory these implementations are much more efficient and can fit millions of p-bits in a single core, leading to a projected 1.000.000 flips per nanosecond [22]. In practice, there still various difficulties to overcome in realizing these devices [23].

As of date, numerous approaches are being explored for implementing hardware p-bits. The emergence of probabilistic computers appears to be not a matter of 'if' but rather 'when'. This scenario is reminiscent of the 1950s when the transistor was invented, yet computers as we know them had not been developed.

Table 1: Results from Nianzi et. al. (2023). Comparison of a FPGA-based MCMC sampler with standard CPU and graph-colored CPU implementations.

Sampling method	topology	size	max. degree	flips/ns
Standard Gibbs (CPU)	Pegasus	4,264	15	2.18×10^{-5}
GC Gibbs (CPU)	Pegasus	4,264	15	8.50×10^{-3}
GC Gibbs (FPGA)	Pegasus	4,264	15	6.40×10^{1}
Standard Gibbs (CPU)	Zephyr	3,360	20	2.67×10^{-5}
GC Gibbs (CPU)	Zephyr	3,360	20	4.40×10^{-3}
GC Gibbs (FPGA)	Zephyr	3,360	20	5.04×10^{1}

Table 2: Results from Aadit et. al. (2022). Comparison of the FPGA-based and nanodevice-based (projected) stochastic Ising machine (sIM) with optimized GPU and TPU implementations of Markov Chain Monte Carlo sampling. It is important to note that the highly-optimized GPUs and TPUs are solving simple, nearest-neighbor chessboard lattices, unlike the irregular and relatively high-degree (with up to 20 neighbors) graphs used for the FPGA implementations.

Platform	Graph	flips/ns
Nvidia Tesla C1060 GPU	Chessboard	7.98
Nvidia Tesla V100 GPU	Chessboard	11.37
Google TPU	Chessboard	12.88
Nvidia Fermi GPU	Chessboard	29.85
FPGA sIM	Irregular	143.80
Nanodevice sIM [Projected]	Irregular	1,000,000

References

[1] D. J. Griffiths and D. F. Schroeter, *Introduction to quantum mechanics*. Cambridge university press, 2018.

[2] A. Flores-Riveros, N. Aquino, and H. Montgomery Jr, "Spherically compressed helium atom described by perturbative and variational methods," *Physics Letters A*, vol. 374, no. 10, pp. 1246–1252, 2010.

[3] D. Reisinger, R. Adam, M. L. Kogler, M. Füllsack, and G. Jäger, "Critical transitions in degree mixed networks: A discovery of forbidden tipping regions in networked spin systems," *PloS one*, vol. 17, no. 11, e0277347, 2022.

[4] R. J. Baxter, *Exactly solved models in statistical mechanics*. Elsevier, 2016.

[5] M. Mézard, G. Parisi, and M. A. Virasoro, *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*. World Scientific Publishing Company, 1987, vol. 9.

[6] X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural networks," *Nature communications*, vol. 8, no. 1, p. 662, 2017.

[7] Y. Nomura, "Boltzmann machines and quantum many-body problems," *Journal of Physics: Condensed Matter*, vol. 36, no. 7, p. 073 001, 2023.

[8] G. Carleo, Y. Nomura, and M. Imada, "Constructing exact representations of quantum many-body systems with deep neural networks," en, *Nature Communications*, vol. 9, no. 1, p. 5322, Dec. 2018, Number: 1 Publisher: Nature Publishing Group, ISSN: 2041-1723. DOI: 10.1038/s41467-018-07520-3. [Online]. Available: https://www.nature.com/articles/s41467-018-07520-3 (visited on 10/18/2023).

[9] R. P. Feynman, "Space-time approach to non-relativistic quantum mechanics," *Reviews of modern physics*, vol. 20, no. 2, p. 367, 1948.

[10] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100 379, 2021.

[11] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: Methods and implementations," *Reports on Progress in Physics*, vol. 83, no. 5, p. 054 401, 2020.

[12] W. R. Clements, J. J. Renema, Y. H. Wen, H. M. Chrzanowski, W. S. Kolthammer, and I. A. Walmsley, "Gaussian optical ising machines," *Physical Review A*, vol. 96, no. 4, p. 043 850, 2017.

[13] A. Zegarac and F. Caravelli, "Memristive networks: From graph theory to statistical physics," *Europhysics Letters*, vol. 125, no. 1, p. 10 001, 2019.

[14] J. Kaiser and S. Datta, "Probabilistic computing with p-bits," *Applied Physics Letters*, vol. 119, no. 15, 2021.

[15] S. Niazi, N. A. Aadit, M. Mohseni, S. Chowdhury, Y. Qin, and K. Y. Camsari, "Training deep boltzmann networks with sparse ising machines," *arXiv preprint arXiv:2303.10728*, 2023.

[16] S. Chowdhury, A. Grimaldi, N. A. Aadit, *et al.*, "A full-stack view of probabilistic computing with p-bits: Devices, architectures and algorithms," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 9, no. 1, pp. 1–11, Jun. 2023, arXiv:2302.06457 [physics], ISSN: 2329-9231. DOI: 10.1109/JXCDC.2023.3256981. [Online]. Available: http://arxiv.org/abs/2302.06457 (visited on 10/18/2023).

[17] V. Choi, "Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design," *Quantum Information Processing*, vol. 10, no. 3, pp. 343–353, 2011.

[18] C. S. Von Bartheld, J. Bahney, and S. Herculano-Houzel, "The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting," *Journal of Comparative Neurology*, vol. 524, no. 18, pp. 3865–3895, 2016.

[19] E. Pelofske, "Comparing three generations of d-wave quantum annealers for minor embedded combinatorial optimization problems," *arXiv preprint arXiv:2301.03009*, 2023.

[20] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011.

[21] N. A. Aadit, A. Grimaldi, M. Carpentieri, *et al.*, "Massively parallel probabilistic computing with sparse ising machines," *Nature Electronics*, vol. 5, no. 7, pp. 460–468, 2022.

[22] K. Lee, J. Bak, Y. Kim, *et al.*, "1gbit high density embedded stt-mram in 28nm fdsoi technology," in *2019 IEEE International Electron Devices Meeting (IEDM)*, IEEE, 2019, pp. 2–2.

[23] R. Rahman and S. Bandyopadhyay, "Variability of binary stochastic neurons employing low energy barrier nanomagnets with in-plane anisotropy," *arXiv preprint arXiv:2108.04319*, 2021.